International Journal of Geographical Information Science
Vol. 26, No. 5, May 2012, 771–793

Taylor & Francis
Taylor & Francis Group

Extended hybrid meshing algorithm for multiresolution terrain models

E.G. Paredes[a]*, M. Bóo[a], M. Amor[b], J.D. Bruguera[a] and J. Döllner[c]

[a]*Department of Electronics and Computer Science, University of Santiago de Compostela, Santiago de Compostela, Galicia, Spain;* [b]*Department of Electronics and Systems, University of A Coruña, A Coruña, Galicia, Spain;* [c]*Hasso-Plattner-Institut, University of Potsdam, Potsdam, Germany*

(*Received 14 December 2010; final version received 26 July 2011*)

Hybrid terrains are a convenient approach for the representation of digital terrain models, integrating heterogeneous data from different sources. In this article, we present a general, efficient scheme for achieving interactive level-of-detail rendering of hybrid terrain models, without the need for a costly preprocessing or resampling of the original data. The presented method works with hybrid digital terrains combining regular grid data and local high-resolution triangulated irregular networks. Since grid and triangulated irregular network data may belong to different datasets, a straightforward combination of both geometries would lead to meshes with holes and overlapping triangles. Our method generates a single multiresolution model integrating the different parts in a coherent way, by performing an adaptive tessellation of the region between their boundaries. Hence, our solution is one of the few existing approaches for integrating different multiresolution algorithms within the same terrain model, achieving a simple interactive rendering of complex hybrid terrains.

Keywords: 3D modeling; 3D visualization; geovisualization; triangulated irregular networks

1. Introduction

Interest in digital terrain models has been increasing over the last decade, in both academic and industrial domains. More powerful computers and generalized use of visualization software have introduced interactive rendering of terrains in a broad range of areas of application. Geographic information systems, urban planning, geological and environmental sciences, training simulators, virtual environments and video games are some of the most relevant.

However, real-time rendering of complex terrain models is not trivial and it poses several challenges. Despite the development of graphics hardware, rendering capabilities of modern workstations are not always high enough to deal with the most detailed terrain data sets, which are continuously incorporating more accurate data. Most of the issues can be minimized by rendering the mesh using level-of-detail (LOD) techniques. Several LOD approaches can be found in literature, most of them based on exploiting temporal coherence (Duchaineau *et al.* 1997), restricted triangulations (Pajarola 1998), progressive meshes (Hoppe 1997; Hoppe 1998) or hierarchies of simplified patches (Levenberg 2002;

*Corresponding author. Email: eg.paredes@usc.es

ISSN 1365-8816 print/ISSN 1362-3087 online

http://dx.doi.org/10.1080/13658816.2011.615317
http://www.tandfonline.com

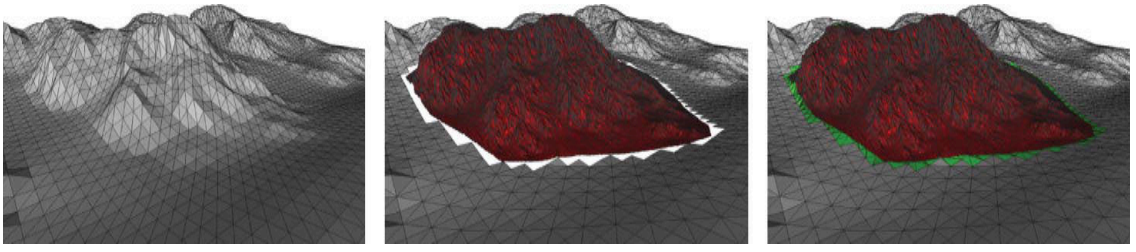
Schneider and Westermann 2006). Exhaustive surveys on the rendering of multiresolution terrains, for example, DeFloriani *et al.* (1996); Luebke *et al.* (2002); Pajarola and Gobbetti (2007), are available in the literature.

Digital terrain models are represented using different data structures depending on several factors, such as the data acquisition technology and the domain of application. The digital elevation model (DEM) stores the geometry information in a uniform equispaced elevation grid, which can be easily encoded using only the elevation values for each position of the grid. Another commonly used model in the representation of digital terrains is the triangulated irregular network (TIN), which stores data as a triangular mesh formed by the tessellation of irregularly distributed points in the terrain. Hybrid models integrate information with different data representations in the same model. In real-world situations, it is not uncommon to find several models for the same terrain area. Hybrid models may enhance existing grid-based models by adding details to specific regions, without increasing the overall grid resolution nor modifying the existing terrain models. In the CityGML standard (Gröger *et al.* 2008) several examples of integration between local TIN models and larger regular ones can be found.

However, the direct visualization of hybrid models would lead to visual discontinuities and overlapping triangles, since mesh data arrive from different datasets and vertices are disconnected. Thus, both meshes should be locally adapted to avoid artifacts in the junction of the meshes as pointed out in Baumann *et al.* (1999) and Yilmaz *et al.* (2004). In hybrid models formed by a regular base grid and some finer resolution TINs, the adaptation may be performed by an adaptive tessellation of the grid cells (Figure 1a) where the boundaries of the TIN models lay (Figure 1b and c). Adding multiresolution to those kinds of hybrid terrains is even more complicated, since component meshes change dynamically depending on the viewing conditions. Relatively similar problems have been studied in other domains, as 3D surface reconstruction from contours in medical imaging (Fuchs and Kedem 1977) or geographic information systems (Chai *et al.* 1998, Hormann *et al.* 2003). However, those solutions are usually not feasible in real-time domains, especially when the boundaries are dynamic, as in the case of a multiresolution terrain rendering application.

In Yang *et al.* (2005), a working method was presented based on the generation of a new preprocessed multiresolution mesh. This article proposes a method to construct multiresolution models in order to combine grid and TIN meshes. A representation of integrated terrain models and a methodology for the construction of integrated multiresolution terrain models based on a set of rules are proposed. However, in this approach a new model is created from the union of the grid and TIN meshes for each LOD.

A different approach that does not alter the original data, the hybrid meshing (HM) algorithm, was presented in Bóo *et al.* (2007), and later modified in Bóo and Amor (2009)



(a) (b) (c)

Figure 1. HM algorithm example. (a) Base regular mesh. (b) Base regular mesh and detailed TIN mesh. (c) Single coherent mesh obtained by the HM algorithm.

to permit the dynamic modification of the TIN position. The HM algorithm joins both models in real-time through a local, cell-based strategy (see Figure 1c). During an initialization phase, appropriate tessellations for joining every LOD of the grid with the TIN model are computed and encoded. Then, during visualization, the triangles required for the selected grid LOD are decoded on demand from this unified representation. The final hybrid model is formed by the union of the view-dependent refinement of the grid and the TIN model, linked in a single coherent mesh.

In this article, we present a new extended hybrid meshing (EHM) algorithm that we have developed to add support for the rendering of multiresolution TIN and grid models. This poses a significant challenge, since using a multiresolution TIN model implies the dynamic simplification of the TIN boundary (TB), which is indeed a key structure in the tessellation process of the original HM algorithm. Our new proposal is based on the same approach of the original method, that is, encoding the precomputed local tessellations for any LOD of TIN and grid. Furthermore, the new tessellation algorithm successfully detects which boundary vertices are active in the selected TIN LOD and readjusts the generation of the tessellation appropriately. This dynamic reconfiguration of the precomputed tessellation is possible due to some convenient restrictions in the order of elimination of the TB vertices, enforced during the preprocessing. Therefore, our proposal retains most of the interesting properties of the original HM algorithm, such as the small memory overhead, high-quality local tessellation and preservation of the original data without alterations. Moreover, since in our solution the grid and TIN parts are dynamically simplified, depending upon the view point, the resulting hybrid model preserves the terrain details while it reduces the number of rendered triangles.

This article is structured as follows: Section 2 summarizes the original HM algorithm. Section 3 introduces our proposal to include a multiresolution model in the TIN part. Section 4 analyzes the requirements of the multiresolution method employed on the TIN mesh to ensure the coherence of the final mesh. Section 5 presents some test results and, finally, Section 6 presents the conclusions of our work.

2. Hybrid meshing algorithm

The HM algorithm is a method for rendering hybrid terrains introduced in previous publications (Bóo *et al.* 2007; Bóo and Amor 2009). It is capable of rendering multiresolution hybrid models formed by high resolution TIN meshes embedded in a multiresolution regular grid model. Furthermore, it uses an efficient data representation scheme which is not dependent on the selected LOD of the model. The algorithm works following a local strategy to perform an adaptive tessellation between the borders of the different parts of the model.

During rendering, the HM algorithm computes the LOD in the grid mesh, depending on the viewing conditions and the regular multiresolution algorithm employed. Once this has been done, grid cells not covered with the TIN (*NC cells*) are sent to the rendering pipeline while the completely covered grid cells (*CC cells*) are discarded, as they will be replaced by the more detailed TIN data. The partially covered grid cells (*PC cells*), where the borders of the TIN lie, are adaptively tessellated. An example of grid cells classification is shown in Figure 2a in orthogonal projection.

To perform the tessellation of the PC cells, the HM algorithm uses a two-step procedure consisting of the local convexification of the TB, and the generation of triangles to join the local convex hulls of the TB and the grid cells. The convexification of the TB is precomputed and encoded in a lightweight data structure during an initialization phase. Thus,

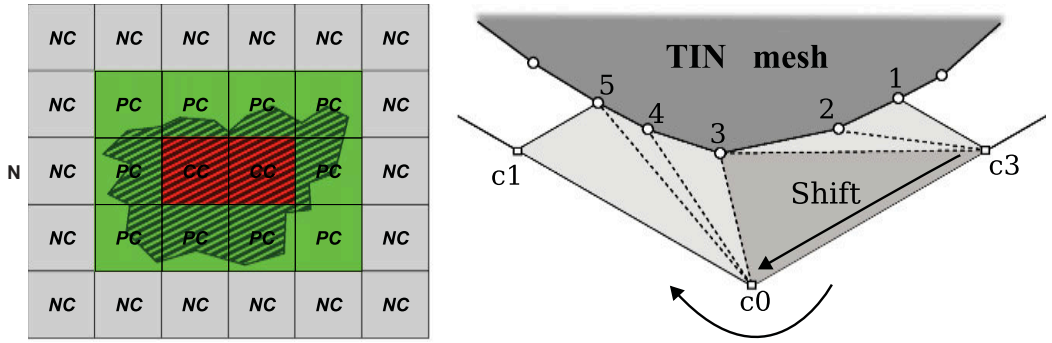


Figure 2. HM algorithm. (a) Grid cells classification. (b) Cell tessellation.

convexification triangles can be generated during rendering by means of simple decoding operations. Once the TB has been convexified, TB vertices can easily be connected to the grid cell corners. In the following subsections the main steps of the algorithm are briefly summarized. For reasons of clarity, this list process is explained first.

2.1. Tessellation of the partially covered grid cells

During the hybrid model visualization, the TIN and grid meshes are joined without gaps owing to the local tessellation of the PC grid cells. The tessellation algorithm starts connecting the vertices of the locally convexified TB with an uncovered corner of the cell. Then, it continues linking the cell corner with consecutive vertices of the TB while the introduced triangle does not overlap with the TIN. When this happens, a *shift* in clockwise manner to the next cell corner is made before continuing with the triangulation.

Figure 2b shows an example where a PC cell and the corresponding convex TIN silhouette are depicted in isometric projection. The TB vertices (1,2,3,4,5) and the cell corners uncovered by the TIN ($c0, c1, c3$) are processed in a clockwise manner. The HM algorithm connects the list of corners with the list of vertices, following a sequential order. In this example, corner $c3$ is connected with vertices 1, 2 and 3, but not with 4 because the new triangle ($c3$,3,4) would overlap with the TIN. Consequently, corner $c0$ is selected and connected with 3 and 4.

2.2. Incremental convexification generation

It has been pointed out above that a local convexification of the TB is performed during the tessellation of the PC cells. However, as this is a costly operation, the TB convexification is actually precomputed and encoded for each PC cell of the grid mesh.

The convexification is performed incrementally, starting from the finest LOD of the grid. The process consists in the identification and tessellation of the *caves*, that is, the concave parts of the TB, for each cell. Note that any triangulation algorithm may be used for this task, as the HM algorithm imposes no restrictions on the triangulation of the detected caves. Next, the following coarser levels are processed preserving the triangles generated during the convexification of previous levels. This incremental procedure continues for each LOD in the grid, until the coarsest one is completed.

By way of example, Figure 3a shows four cells corresponding to the finest resolution level of a grid and the corresponding TIN silhouette covering the area. The TIN is depicted

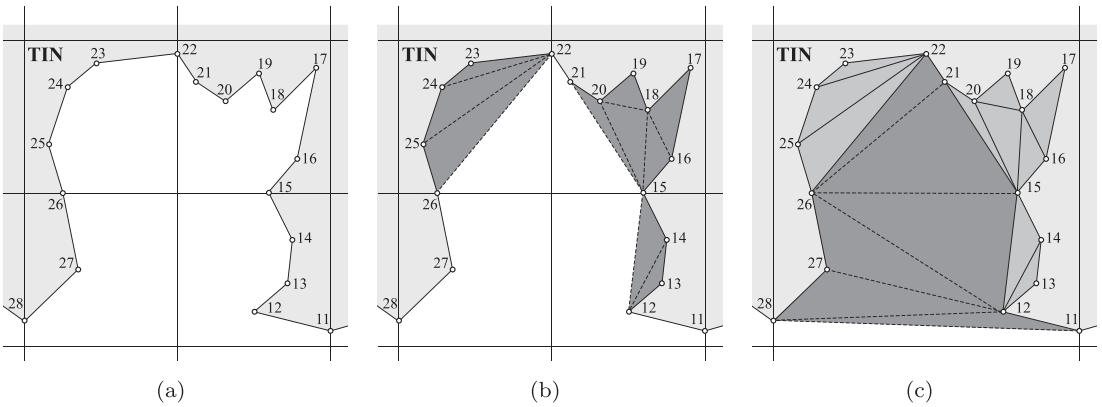


Figure 3. Incremental convexification of the TIN boundary. (a) Original cells. (b) Finest LOD convexification. (c) Next LOD convexification.

in gray while the TIN boundary vertices are represented with circles. In this example, local convex hulls are delimited by vertices {11, 12, 15} (lower-right cell), {15, 21, 22} (upper-right cell), {22, 26} (upper-left cell) and {26, 27, 28} (lower-left cell). The triangulation of the caves after preprocessing the finest LOD are displayed in Figure 3b. In Figure 3c, the following coarser LOD is analyzed and the new local convex hull is determined by vertices {11, 28}. New convexification triangles are shown in a different color to the ones generated in the previous step.

2.3. Hybrid model representation

The HM algorithm uses a set of lists to encode the information. In this section, we briefly summarize the TB list as the EHM is based on the modification of this list in order to achieve multiresoution TIN. The full detailed description of the data encoding can be found in Bóo *et al.* (2007).

The TB convexification data are encoded as a list of the TB vertices and their additional connectivity information. Since the TB is stored following a clockwise ring structure, the connectivity associated to each vertex indicates the distance, that is, the number of vertices between that vertex and the most distant one in the ring connected to it. Thus, if the connectivity of vertex i is j, this means that the farthest vertex in the ring connected to it is vertex $i + j$. This connectivity value is the only additional piece of information to be stored per boundary vertex.

Let us consider again the example described in Figure 3c to illustrate this storing strategy. The TB corresponding to the TIN depicted in that figure can be represented with the TB list:

$$TB = \{ \ldots \ 11(17), 12(16), 13(1), 14(1), 15(11), 16(2), 17(1), 18(2), 19(1), \\ 20(1), 21(5), 22(4), 23(1), 24(1), 25(1), 26(1), 27(1) \ \ldots \} \quad (1)$$

where connectivity value of each vertex is indicated within brackets. For example, vertex 15, with a connectivity value of 11, is connected with vertex 26 and all the vertices between them not placed inside a nested cave. Nested caves are simply detected by looking for vertices with connectivity values greater than one, which indicates that the vertex is the beginning of a cave. In this case, connectivity values indicate three caves: between

vertices 16 and 18, between vertices 18 and 20 and between vertices 21 and 26. The algorithm assumes a sequential connection of the cave starting vertex to the following vertices until the cave ending, but this connecting structure is broken if nested caves exist. For this example, vertex 15 is connected to all vertices between 16 and 26 not belonging to a nested cave, that is, {16, 18, 20, 21, 26}.

As explained in Bóo *et al.* (2007) the connectivity values of the coarsest LOD can be used for any other LOD. Hence, this unified representation of the convexification triangles may be employed for the tessellation of the caves at any level of detail of the grid.

3. Extended HM algorithm

In the HM algorithm, LOD techniques are exploited on the base grid, while TIN models are used to add supplementary details to especially relevant areas. However, the number of rendered primitives is unnecessarily incremented by processing the full detailed TIN models in conditions where such highly detailed models are unnecessary. In these situations, for example, when the TIN model is placed far from the viewpoint, the number of vertices processed in the TIN mesh will be disproportionate.

On the other hand, HM algorithm is a valuable approach for visualizing hybrid terrains, as has been analyzed in Paredes *et al.* (2009). It is simple, efficient, inherently flexible – due to the independence of the multiresolution method used in the regular part of the model – and obtains good quality tessellations on the dynamically generated mesh. Consequently, the EHM algorithm has been developed to overcome the limitation of using single-resolution TIN models, introducing a new solution to deal with dynamically simplified TIN models. Since it is largely based on the HM algorithm, it deliberately adopts the notation and terminology defined in HM algorithm references (Bóo *et al.* 2007; Paredes *et al.* 2009).

In the following subsections, the EHM algorithm is presented. First, the overall strategy of the method is discussed and then the new tessellation algorithm, which dynamically generates a coherent drawable model, is explained in greater detail.

3.1. Algorithm structure

The key point in the original HM algorithm is the preprocessing of the convexification triangles for every grid level, encoded in a unified representation: the TB list. The objective of the EHM algorithm is to extend the capabilities of the TB list to permit the rendering of multiresolution TIN and grid models.

Including a multiresolution TIN in the EHM framework entails dynamic changes in the grid and TIN boundaries during interactive visualization. Thus, only the TB vertices existing in the selected LOD of the TIN mesh are rendered. In this case, a TB substantially different from the full-resolution version will be generated, complicating the tessellation procedure between the models.

If all the vertices in the original TB list are used for generating the convexification triangles, undesired overlapping artifacts will appear. By way of example, Figure 4 shows the result of the original tessellation procedure when applied to a multiresolution TIN. Figure 4a replicates the example of Figure 3 that corresponds to the high-resolution TIN. In this case, the original TIN is depicted in lighter tone and the triangles associated with the convexification procedure, encoded in the TB list of Equation 1, in a darker one. Figure 4b shows the change in shape for a different TIN resolution. In this figure, inactive vertices (i.e. vertices that are not visible in the current LOD of the TIN) have been crossed out

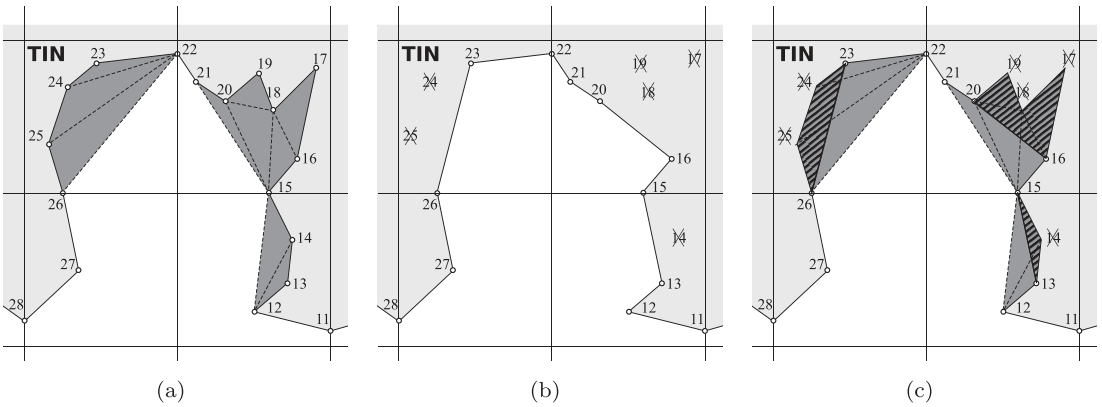


Figure 4. Example of tessellation overlapping problems with a multiresolution TIN boundary. (a) Original finest resolution TIN boundary. (b) Partially simplified TIN boundary. (c) Overlapping between original convexification triangles and partially simplified TIN boundary.

and the new TIN shape is presented in light tone. Finally, Figure 4c shows the overlapping problems that appear if the HM algorithm is directly applied to this TIN resolution level. Following the original convexification instructions encoded in the TB list, invalid triangles containing inactive TB vertices will overlap the TIN mesh. Therefore, proceeding with the original tessellation strategy produces a model with meshing artifacts around the vertices which have been removed from the TB in the selected LOD.

Our solution combines precomputed multiresolution TINs and dynamic tessellation of the boundaries to discard the invalid convexification triangles generated from the original TB list. In a similar spirit to the HM method, the strategy basically comprises two phases: preprocessing and visualization. The main steps of the EHM algorithm are:

- *Preprocessing:* During this phase, which occurs only once at the beginning of the process, the data structures needed by the EHM and the multiresolution algorithms are initialized in the following order:
  (1) Construction of auxiliary LOD structures used by the grid.
  (2) Precomputation of the EHM lists such as the TB list.
  (3) Construction of TIN multiresolution hierarchy which is required by most multiresolution TIN methods (Hoppe 1997; El-Sana and Varshney 1999; Kim and Lee 2001). Since this hierarchy must comply with the simplification preconditions required by the EHM algorithm, it needs to be created after the EHM lists, as will be explained in detail in Section 4.
- *Rendering:* For each rendered frame, the following processing steps are performed:
  (1) Extract view-dependent grid: the grid multiresolution method is used for the extraction of a new LOD according to the current viewing parameters.
  (2) Extract view-dependent TIN: again, using the standard multiresolution method applied in the TIN, a new LOD is extracted for the TIN part.
  (3) Render visible NC grid cells belonging to the computed LOD.
  (4) Render the computed TIN LOD (corresponding to the CC cells area).
  (5) Tessellate and render PC cells using the EHM tessellation algorithm to join both LOD models. The EHM tessellation algorithm is carefully designed to generate the appropriate triangles for the extracted grid and TIN LODs. Therefore, information in the TB list is not blindly processed to generate triangles, but interpreted according to the refined models.

Hence, the EHM algorithm incorporates view-dependent rendering of the TIN parts and can be regarded as a new solution for achieving a full multiresolution hybrid model. Section 3.2 presents the PC cells tessellation algorithm, which corresponds to the final step in the rendering phase.

3.2. Tessellation algorithm for the partially covered grid cells

The EHM algorithm is based on local tessellations by means of the real-time decoding of precomputed data structures. Using multiresolution TINs, however, means that dynamic modifications will occur in the TB. Therefore, a new tessellation algorithm supporting dynamically simplified TB lists has been developed and is presented in this section.

This new EHM tessellation algorithm generates the two kind of triangles used in the PC cells tessellation: convexification triangles filling the caves in the TB convex hull, and corner tessellation triangles effectively linking the TIN convex hull with the grid cell corners.

In the incremental convexification procedure, the vertices of every triangle generated inside a cave present the form:

$$\{caveStart, i, i + (vertex\ i).conn\} \tag{2}$$

where *caveStart* is the TB index of the cave starting point and i corresponds to the index of any inner vertex of the cave, and thus $caveStart < i < caveEnd$. The third term (vertex i).conn represents the connectivity value of the i-th TB vertex. Based on this property, a proper convexification of the TB is achieved for any active resolution level in the TIN mesh. In this way, two cases can be considered for each TB encoded convexification triangle, according to the currently active vertices on the TB: skip the triangle generation or continue with the generation adapted as needed to the TB. Triangles skipped are those whose first or second vertex index, following the TB list order as in Equation (2), are inactive in the current TB. Those triangles are not needed because they are connecting an inactive vertex and instead of filling some area inside a TB cave, they can actually overlap the current TIN mesh. However, when the only inactive vertex in a triangle is the last one $i + (\text{vertex } i).conn$, the triangle is needed to fill the cave and thus it is generated using the next active vertex in TB as ending vertex. Note that the indices of vertices in TB list are maintained throughout the entire process, to avoid invalidating the connectivity-based scheme of the tessellation. Additionally, it is guaranteed that any view-dependent TB computed by the TIN multiresolution algorithm will be compatible with this scheme, due to the preconditions imposed in the construction of the multiresolution hierarchy, which ensures that TIN vertices are removed in a compatible order (see Section 4.2).

In addition to the TB convexification triangles, the tessellation algorithm also generates corner tessellation triangles linking the TIN mesh with the grid cell corners. These triangles follow the scheme mentioned above in Section 2.1, that is, they are sequentially connected to the first uncovered cell corner while the generated triangle does not overlap the mesh. When overlapping is detected, the next cell corner is used to generate the triangle until a new overlapping arises or the process finishes.

An outline of the tessellation algorithm processing each PC cell of the model is shown in Figure 5. Basically, the algorithm iterates through the TB vertices contained in the cell and generates the required triangles as needed. Some auxiliary lists to maintain cave-related information are initialized at the beginning and conveniently updated in each iteration,

```
Require: 0 ≤ N < TB.size and L > 0
    {N = index of the first cell vertex in TB}
    {L = total number of vertices in cell}
 1: k ← 0 {counter of currently opened caves}
 2: startL ← [ ] {list of cave starting points}
 3: endL ← [ ] {list of cave ending points}
 4:
 5: for all i such that N ≤ i < N + L do
 6:    {– Skip processing for inactive vertices –}
 7:    if vertex i is not active then
 8:       continue
 9:    end if
10:    {– Update counters for this iteration –}
11:    while k > 0 and i ≥ endL_k.conn do {Determine if cave ending has been reached}
12:       k ← k − 1
13:    end while
14:    next ← i + 1 {Look for next active vertex}
15:    while vertex next is not active do
16:       next ← next + (vertex next).conn
17:    end while
18:    {– Geometry generation –}
19:    if k = 0 or i + (vertex i).conn ≥ N + L  then {Process convex hull vertices}
20:       LinkToCellCorners(vertex i)
21:    else {Process cave inner vertex}
22:       MakeTriangle(startL_k, vertex i, vertex next)
23:    end if
24:    {– Update counters for next iteration –}
25:    last ← i + (vertex i).conn
26:    if (vertex i).conn > 1 and (vertex last) is active then
27:       {Init new cave if ending is active}
28:       k ← k + 1
29:       startL_k ← i
30:       endL_k ← i + (vertex i).conn
31:    end if
32: end for
```

Figure 5. Tessellation algorithm for the partially covered grid cells.

before and after the geometry generation stage. In this geometry stage, only one kind of convexification or corner tessellation triangle is generated in a single iteration. Examining the algorithm in detail, the first step is to verify that the current iteration vertex *i* is active (line 7), otherwise the processing is skipped. Then, if vertex *i* is found to be an endpoint of a previously opened cave in the local TIN convex hull (line 11), the opened caves counter *k* is updated. Next comes the actual geometry generation step, where a new convexification triangle is emitted (line 22) only when it is needed. This condition is determined by checking whether the vertex belongs to the cell convex hull (line 19), which means that the cave is not active. In this case, the vertex is processed as a convex hull point and thus the required corner tessellation triangles are emitted to link the point to the cell corners, according to algorithm explained in Section 2.1; otherwise the convexification triangle is

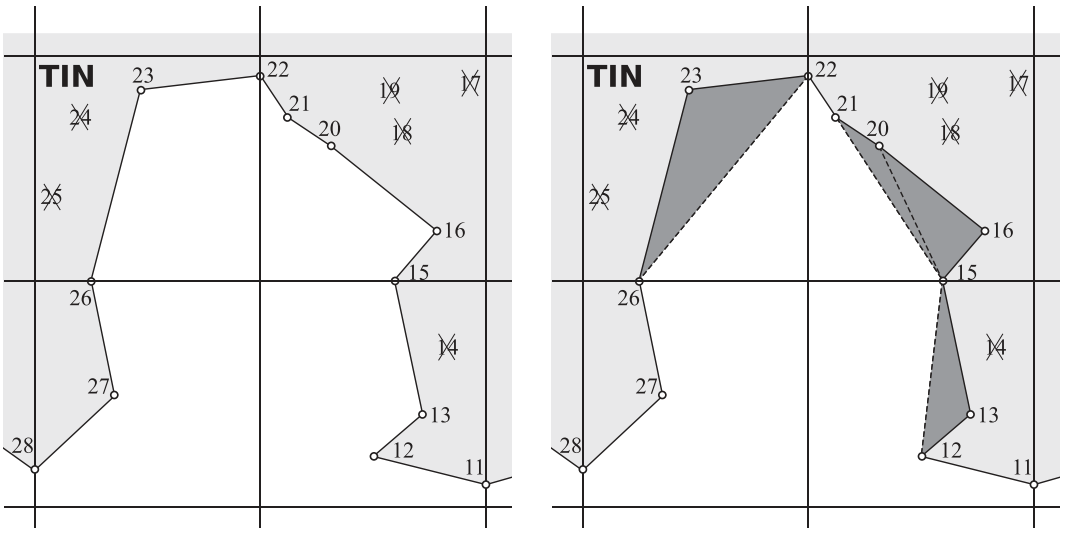


Figure 6. Tessellation algorithm example using a multiresolution TIN boundary. (a) Active TIN LOD. (b) Convexification triangles generated by the tessellation algorithm in the active TIN LOD.

safely generated. Finally, if the vertex also marks the beginning of a new or a nested cave (line 25), the cave beginning and ending indices are stored in *startL* and *endL* lists, and the opened cave counter k is increased. It should be noted that caves are always considered as opened, even if the ending point belongs to a different cell and that cave is later ignored.

Let us illustrate the operation of the algorithm by considering the tessellation of the partially refined TIN mesh in Figure 4. For convenience, the TIN mesh is pictured again in Figure 6a, while the obtained convexification is represented in Figure 6b. Note that connectivity values of the TB vertices remain unaltered, although some of the original TB vertices are inactive in the current TIN resolution:

$$\begin{aligned} TB = \{\ldots\ & 11(17), 12(16), 13(1), \cancel{14}(1), \\ & 15(11), 16(2), \cancel{17}(1), \cancel{18}(2), \cancel{19}(1), \\ & 20(1), 21(5), 22(4), 23(1), \cancel{24}(1), \\ & \cancel{25}(1), 26(1), 27(1)\ \ldots\} \end{aligned}$$

Table 1 shows the operation of the algorithm during the sequential processing of vertices from 11 to 26 in the TB list, divided into three blocks, corresponding to each one of the grid PC cells. The algorithm avoids meshing problems by ignoring the processing of input vertices or generating tessellation triangles when appropriate. For example, let us follow the processing of first vertices in the example TB list. Vertex 11 is active and is not inside any previously opened cave, thus, it is added to the convex hull. It also has a connectivity value of 17, meaning that a large cave ending in the vertex 28 is opened here; corresponding values are added to the starting (*startL*) and the ending (*endL*) lists. The same processing is performed with vertex 12 and a new nested cave is opened. Then, a new convexification triangle with vertices {12, 13, 15} is generated during the processing of the vertex 13, which has a connectivity value of 1. As the third vertex of the triangle should be {13 + (vertex 13.conn} = 14, which is inactive, vertex 15 is used instead, avoiding any overlapping or meshing problems. Vertex 14 is then ignored, since it is inactive, and the processing continues with vertex 15.

Table 1. Tessellation algorithm operation for the example shown in Figure 6.

Vertex *i*	*k*	*startL*	*endL*	TB Conv. Triangles	TB Convex Hull
11	1	[11]	[28]	–	[11]
12	2	[11, 12]	[28, 28]	–	[11, 12]
13	2	[11, 12]	[28, 28]	(12,13,15)	[11,12]
~~14~~	–	–	–	–	–
15	3	[11, 12, 15]	[28, 28, 26]	–	[11, 12, 15]
15	1	[15]	[26]	–	[15]
16	1	[15]	[26]	(15, 16, 20)	[15]
~~17~~	–	–	–	–	–
~~18~~	–	–	–	–	–
~~19~~	–	–	–	–	–
20	1	[15]	[26]	(15, 20, 21)	[15]
21	2	[15, 21]	[26, 26]	–	[15, 21]
22	3	[15, 21, 24]	[26, 26, 26]	–	[15, 21, 22]
22	1	[22]	[26]	–	[22]
23	1	[22]	[26]	(22, 23, 26)	[22]
~~24~~	–	–	–	–	–
~~25~~	–	–	–	–	–
26	0	[]	[]	–	[22, 26]

Hence, this strategy of ignoring inactive vertices while proceeding with the cell tessellation may deal transparently with simplified vertices in the TIN boundary. In fact, it also reduces the tessellation overhead, since inactive vertices are omitted. This simple tessellation scheme is independent of the TIN and grid multiresolution methods, and works even in complicated scenarios, such as a section of the TB with multiple nested caves and randomly simplified vertices. In fact, part of the tessellation complexity is eliminated during preprocessing phase, imposing some preconditions to the TIN multiresolution model which allow us to simplify the runtime algorithm making convenient assumptions about the simplified TB, as explained in section 4.

4. Multiresolution TIN preprocessing

The EHM method does not rely on any particular multiresolution TIN approach. However, the performance and complexity of the EHM algorithm depends partially on the adaptation of its TB-based approach to the multiresolution methods used in the model, as discussed in this section. The first issue to examine is related to the simplification operators used in the preprocessing of the multiresolution TIN model and it is exposed in Section 4.1. In Section 4.2, the effects for the EHM algorithm of the preprocessed vertices removal order are analyzed, and two preconditions are derived to guarantee that a valid hybrid model is generated in any case.

4.1. Multiresolution TIN algorithm

In the EHM method, as mentioned above, LOD rendering is achieved by using multiresolution models for the grid and TIN meshes. Multiresolution models are generally built by recording a series of simplification operations successively applied on the original geometric data from the terrain dataset. These operations are then organized into a hierarchical data structure, where the coarsest version of the model is at the root, and

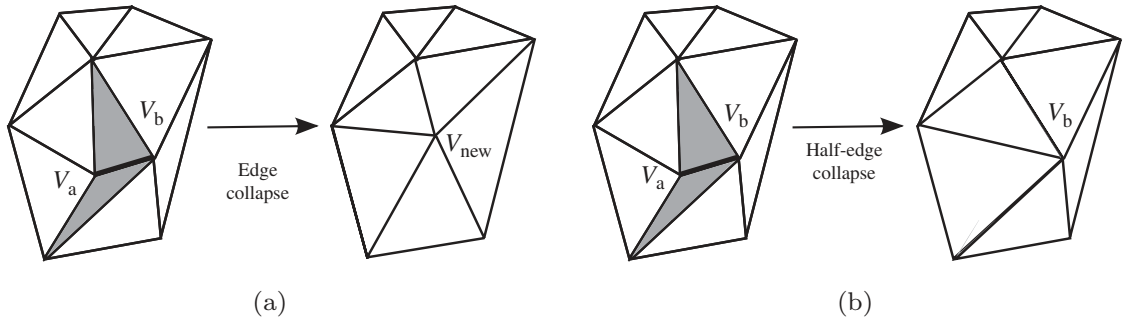


Figure 7. Comparison between simplification operators. (a) Edge collapse. (b) Half-edge collapse.

the fully detailed model in the leaves. Later, during real-time rendering, a view-dependent mesh is extracted from this structure by applying a set of compatible operations. There are many different approaches based on this basic idea (Luebke *et al.* 2002), using different operators and different methods to build and access the multiresolution hierarchy. Some operators simply delete the faces, edges, or vertices of the mesh and do not introduce new elements, while others replace elements of the original model with a simplified version with lower precision but similar characteristics. A comparative example of both types is shown in Figure 7: in Figure 7a a new vertex is introduced with a full edge collapse operator during the simplification of edge $v_a - v_b$; in Figure 7b, using a half-edge collapse operator, the same edge is simplified by collapsing the edge endpoint v_a into the other endpoint v_b, without inserting any new vertex.

In fact, the introduction of new vertices in the TB list gives rise to serious consequences for any TB-based approach, such as the EHM algorithm. For example, Figure 8 depicts a

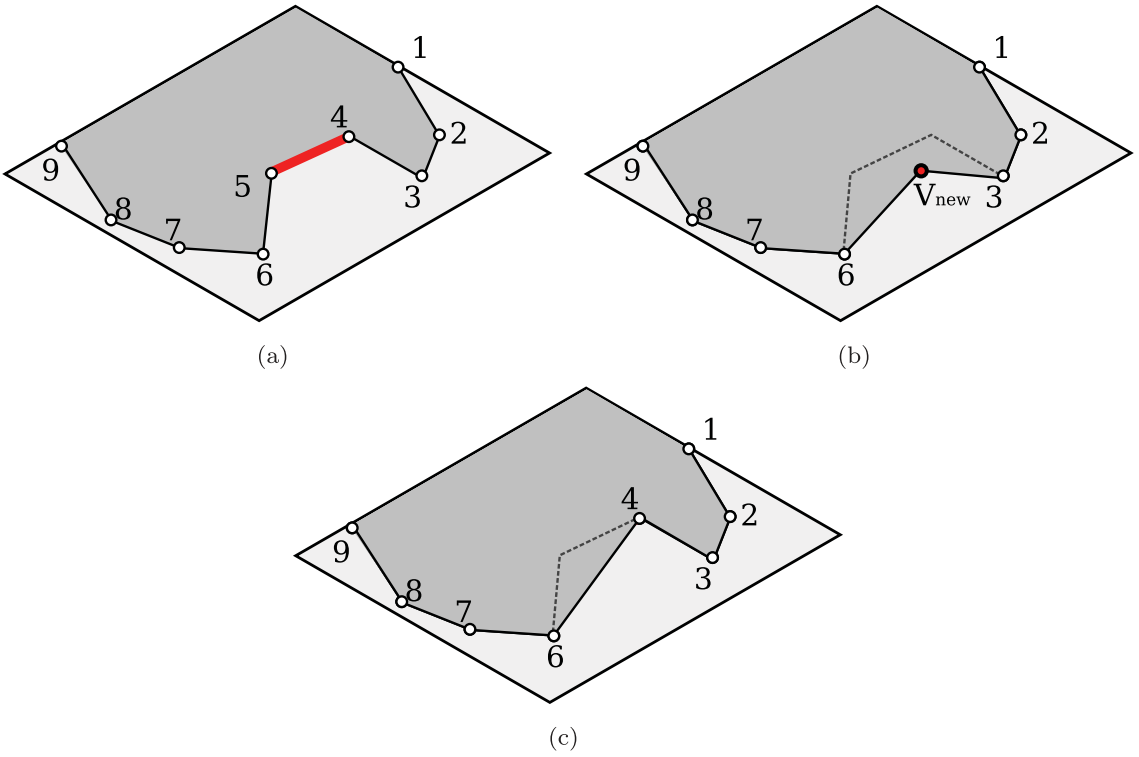


Figure 8. Examples of TIN boundary simplification inside a PC cell. (a) Original TIN boundary. (b) Simplified TIN boundary using an edge collapse operator. (c) Simplified TIN boundary using a half-edge collapse operator.

PC cell where a TB edge inside a PC cell is removed using different operators. The TB information of the original cell in Figure 8a, can be represented as the following array:

$$TB_{\text{original}} = \{1(1), 2(1), 3(3), 4(1), 5(1), 6(1), 7(1), 8(1)\}$$

In Figure 8b the TB edge formed between vertices 4 and 5 is collapsed using a full edge collapse operator, while in Figure 8c the same operation is performed using a half-edge collapse operator. In the first case, a new vertex is introduced in the TB, which would force the updating of the TB list during visualization to reflect this change:

$$TB_{\text{collapsed\_edge}} = \{1(1), 2(1), 3(3), \cancel{4(1)}, \underline{V_{new}(1)}, \cancel{5(1)}, 6(1), 7(1), 8(1)\}$$

In the second case, the half-edge collapse operator proceeds by removing the vertex 5 endpoint of the edge, without introducing new points in the mesh. The resulting TB list would be

$$TB_{\text{half\_collapsed\_edge}} = \{1(1), 2(1), 3(3), 4(1), \cancel{5(1)}, 6(1), 7(1), 8(1)\}$$

The original TB list is not preserved in any case, but using half-edge collapse operators it becomes possible to develop a strategy for ignoring the collapsed vertices during the generation of the convexification triangles, as seen in Section 3.2.

In conclusion, the introduction of new vertices into the original mesh implies that the TB must be completely recalculated during visualization for each frame. Thus, it becomes clear that operators introducing new elements in the boundaries are difficult to integrate with our approach, due to the implied performance penalty and overall complexity increasing. Using half-edge collapse operators, on the other hand, is compatible with our approach. Therefore, any half-edge collapse-based multiresolution algorithm may be used in the TIN mesh, provided that the multiresolution TIN models were built according to the preconditions exposed in this Section 4.2.

4.2. Multiresolution TIN preconditions

As stated above our approach does not enforce the use of any particular method, as long as it uses a half-edge collapse operator during simplification. However, to prevent the appearance of any conflictive cases during the tessellation process, we need to guarantee that the dynamically refined TB complies with certain assumed propositions regarding the vertex removal order. These propositions can be formalized as preconditions in the resulting TIN multiresolution hierarchy.

Those required preconditions only affect the boundary vertices of the TIN and they do not restrain the elimination of any vertex, that is, during the construction of the hierarchy the preconditions only modify collapse operations priority, intentionally delaying or bringing them forward, when it is needed. Using this approach the border of any view-dependent refined mesh transparently preserves the consistency of the EHM algorithm. Basically, when a multiresolution hierarchy is built according to these preconditions, any view-dependent refinement of the boundary assumes an inside-out order: the removal of points within the cave before its endpoints is forced, and nested caves are simplified before parent caves, and so on. The tessellation algorithm is capable of obtaining a well-formed

hybrid model with this simple strategy. The preconditions are presented along with their motivations and reasoning, in the following subsections.

4.2.1. *Cave vertices collapse precondition*

The first step in the tessellation of the PC cells is to generate the convexification triangles of the TB. Following the EHM scheme, new triangles are formed for every cave connecting the inner vertices with the beginning endpoint (the initial vertex of the cave in the TB list order), as it is visible from every other vertex in the cave. Vertices belonging to nested caves are not taken into account, as they are joined to their respective beginning cave endpoint.

When the process finishes, generated triangles cover the caves consecutively from beginning endpoints to ending ones. Therefore, all the convexification triangles rely on the corresponding cave beginning vertex and, if it is removed before intermediate vertices in the cave, it will be impossible to complete the convexification of the remaining part of the cave. Furthermore, if the removed vertex marks the beginning of a nested sub-cave, overlapping and wrong oriented convexification triangles are generated from the beginning endpoint of the parent cave. Figure 9 shows an example grid cell with the corresponding TB list:

$$TB = \{1(12), 2(10), 3(1), 4(4), 5(1), 6(1), 7(1), \\ 8(4), 9(1), 10(1), 11(1), 12(1), 13(1)\}$$

This list indicates that several nested caves appear in the cell TB; the parent cave between vertices 2-8, for example, contains two sub-caves in vertices 2-4 and 4-8.

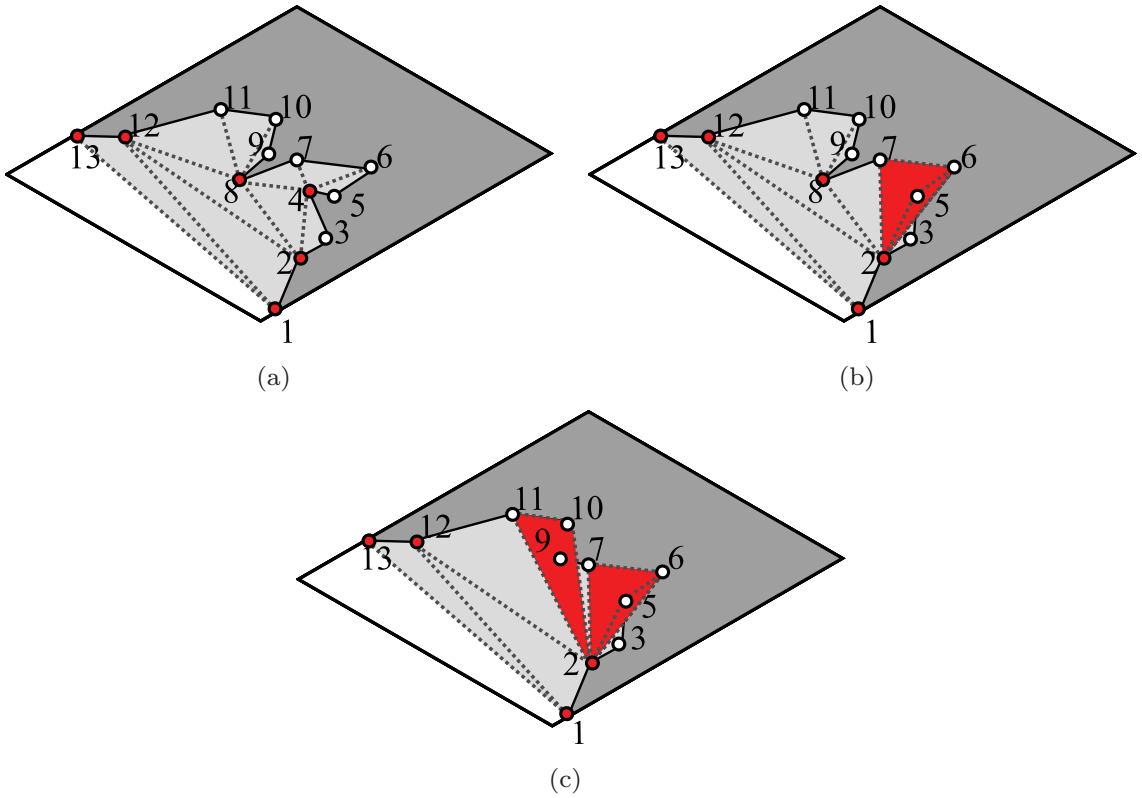


Figure 9. Example of a conflictive simplification of the TIN boundary. (a) Original full-resolution TIN boundary and the associated convexification triangles. (b) Resulting mesh after 4 removal. (c) Resulting mesh after 8 removal.

As previously noted, vertex 4 (having (vertex 4).conn = 4) initiates a simple nested cave ending in 8. These two vertices are the endpoints of the cave, while interior vertices between 5 and 7 are connected only to their next consecutive vertices. If vertex 4 is removed before the interior points of the cave, as indicated in Figure 9b, the origin for the remaining convexification triangles of the cave is lost and hence incorrect overlapping triangles, signaled in red in the figure, are formed from the parent cave beginning, vertex 2 in this case. Removing vertex 8, which marks the beginning of the nested cave in vertices between 8 and 12, leads to a similar situation, as shown in Figure 9c. On the other hand, when the cave endpoints are the last ones to be simplified in the cave, conforming convexification triangles can be generated for any partially simplified cave, as illustrated in Figure 10. In this figure, the original TB with the same previously indicated TB list is depicted in Figure 10a. In Figure 10b, vertices 5,6,7,9,10,11 have been eliminated. With the removal of the interior vertices, the cave between vertices 8 and 12 becomes empty and those endpoint vertices can also be removed if not needed. The next simplification step is shown in Figure 10c, where the cave in vertices 8 and 12 finally disappears since the endpoint vertices 4, 8, and the interior vertex 3 have been removed. After this step vertex 4 is not longer necessary, as the interior vertex 3 is not present in the TB, and the simplification procedure may continue. Figure 10d shows the resulting TB after the removal of vertex 4.

It becomes evident that endpoints are essential for the correct operation of the original HM algorithm, and they must be preserved in the view-dependent refined mesh until all the other vertices of the cave are removed. This requirement can be maintained with no visualization cost, if the relationship is enforced in the multiresolution data structure

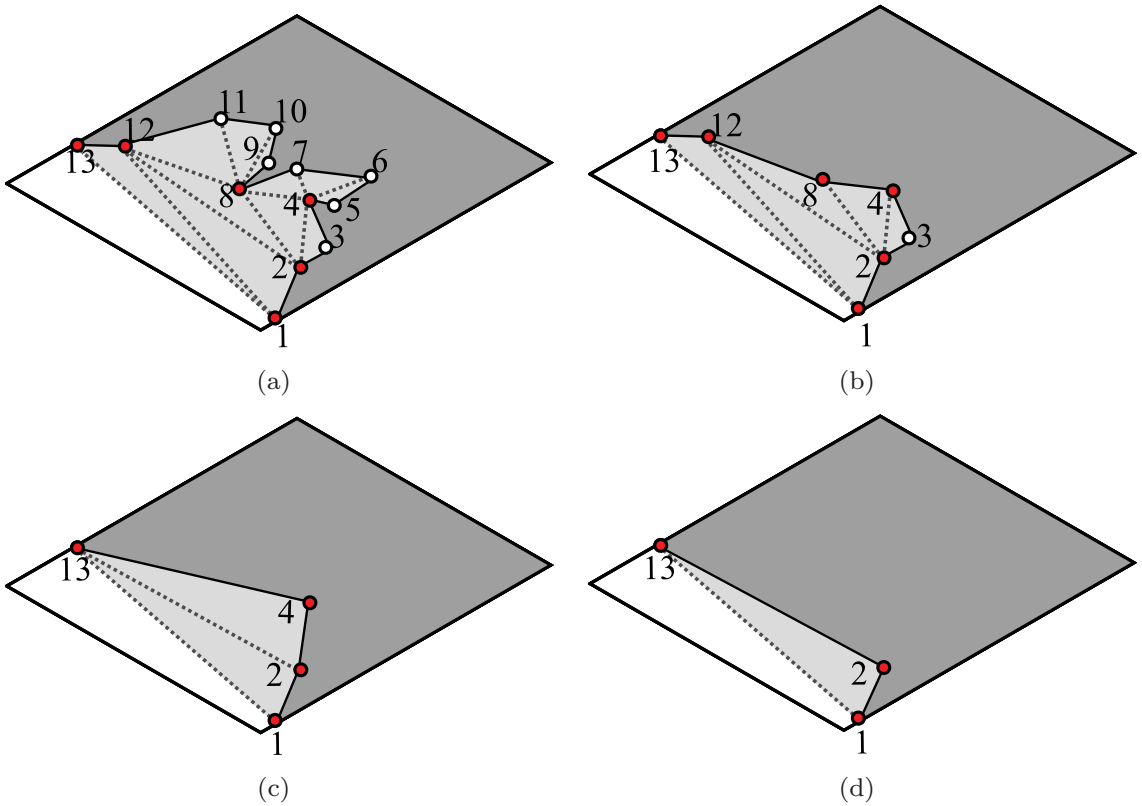


Figure 10. Example of an appropriate simplification of the TIN boundary. (a) Original full resolution TIN boundary and the associated convexification triangles. (b) Resulting mesh after 5, 6, 7, 9, 10, 11 removal. (c) Resulting mesh after 3, 8, 12 removal. (d) Resulting mesh after 4 removal.

computed in the preprocessing phase. Thus, we need to set a precondition stating that whatever irregular multiresolution model is used for the TIN mesh, every interior vertex in a cave must be always simplified before the corresponding endpoints of the cave. Note that this precondition maintains the general flexibility of the algorithm, and the precise removal order for every vertex in the cell is not predefined and still depends on the dynamically changing view conditions.

4.2.2. Cell boundaries preservation precondition

Once the TB has been locally convexified, the second step in the cell tessellation process is the generation of the linking triangles between the borders of both models. As in the previous step, when using a multiresolution TIN model, the vertices of the boundary may be unpredictably simplified during the rendering phase. Although our new tessellation algorithm is capable of handling the absence of some boundary vertices, an additional precondition related to the cell intersection points must be established to avoid the appearance of unsolvable conflicts in the adaptive tessellation.

Tessellation is performed locally on each PC cell of the grid. At a mesh level, TB vertices are introduced at the intersection points with the cell borders to avoid gaps, as in the original HM approach. Since these intersection vertices belong to neighbor cells, the last vertex of the TB array inside a cell is also the first one in the adjacent one, and thus the cell triangulations are transparently joined and the final mesh does not contain any hole.

However, if those intersection vertices are not cave endpoints, they may be removed before other boundary vertices of the cell, leading to the creation of gaps in the resulting mesh. A tessellation example showing gaps between the cells is depicted in Figure 11. In this example, the TIN already has a convex structure inside each cell so that the tessellation can be performed directly by connecting the TB vertices with the uncovered cell corners of that cell (see Section 2.1). The tessellation algorithm will proceed joining uncovered cell corners (squares) with active TB vertices (dots) in a cell-by-cell basis.

In this figure, vertices 1,5,8,11,13 are intersection vertices and, as will be shown below, their elimination could produce undesired results. For the cell delimited by the vertices from 1 to 5, in case (a) as of yet no vertex has been simplified and thus the algorithm will generate triangles linking the three uncovered corners with the TIN convex hull vertices. If vertex 5 is removed, as in case (b), the algorithm will halt the process at vertex 4. Consequently, in the next cell the process will start on vertex 6 and the space falling between vertices 4 and 6 will not be tessellated, giving rise to a hole in the mesh.

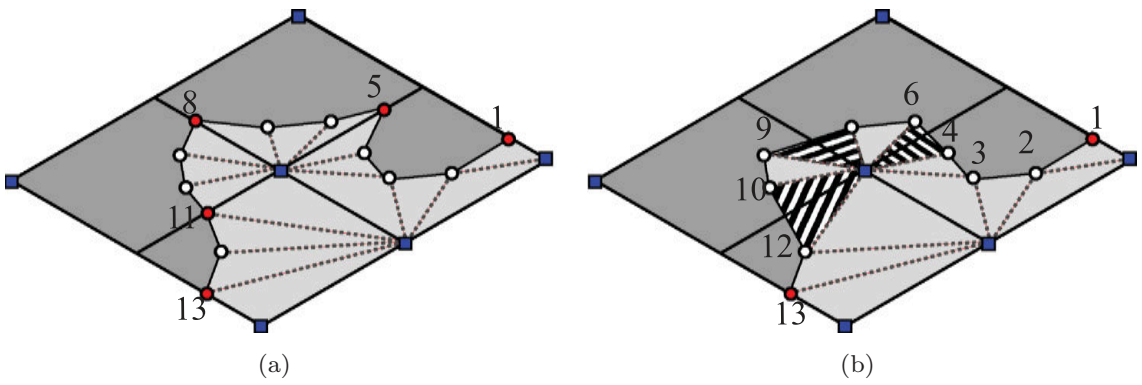


Figure 11. Example of elimination of cell border points. (a) Original mesh. (b) Generation of gaps between neighbor cells.

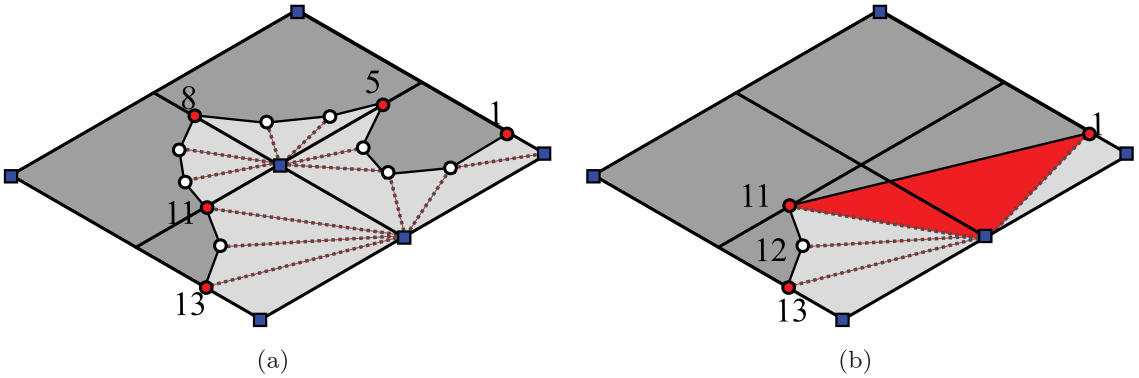


Figure 12. Example of elimination of cell border points. (a) Original mesh. (b) Generation of nonlocal triangles.

An additional problem caused when very different degrees of simplification are computed for the TIN and grid parts should also be considered. During visualization the multiresolution TIN may locally remove a large number of vertices in the TB causing active PC grid cells to be left empty of TIN vertices. In this case, linking triangles between both models should be large enough to cross cell boundaries, in order to tessellate the gap between borders. Nonetheless, our algorithm works locally to every cell, and thus operations involving more than one grid cell are not possible. An example is shown in Figure 12, where a large cross-cell triangle should be generated. Since in Figure 12b all the TB points between vertices 1 and 11 have been simplified, including intersection vertices 5 and 8, a large triangle should be generated linking vertices 1 and 11 with the respective grid cell corner. Note that this is a fairly simple example, but when using real models with irregular borders, nonlocal triangles could be much larger, connecting non-neighbor cells and possibly leading to complex boundaries easily overlapped.

To avoid the nonlocal generation of triangles and holes, a new proposition is derived: cell intersection TB vertices have to be preserved in any view-dependent refinement of the TIN. In our case TIN meshes are considerably more detailed than the base regular grid and many TIN vertices fall in the same grid cell, so this requirement is barely noticeable. Furthermore, TIN meshes are not rendered when very coarse grid LODs are selected for the area of the regular model covered with the TIN, as this implies that the highly detailed geometry added by them is not needed in the current view conditions. Thus, the number of TB vertices needed to preserve the coherence of the final model is small, regardless of which LOD is active in the grid object.

EHM algorithm can be safely used with multiresolution TIN hierarchies fulfilling these two fairly simple preconditions. The conjunction of the real-time tessellation algorithm properties and multiresolution hierarchies built according to the preconditions presented in this section, is enough to guarantee that a well-formed hybrid mesh will be extracted and rendered. In Section 5 we shall analyze some results obtained in our tests with the EHM algorithm.

5. Results

A test application has been developed to validate the EHM algorithm and evaluate the quality of the hybrid models generated. The application, programmed in C++ language, uses OpenGL as the graphic API and the OpenSceneGraph library for the interactive

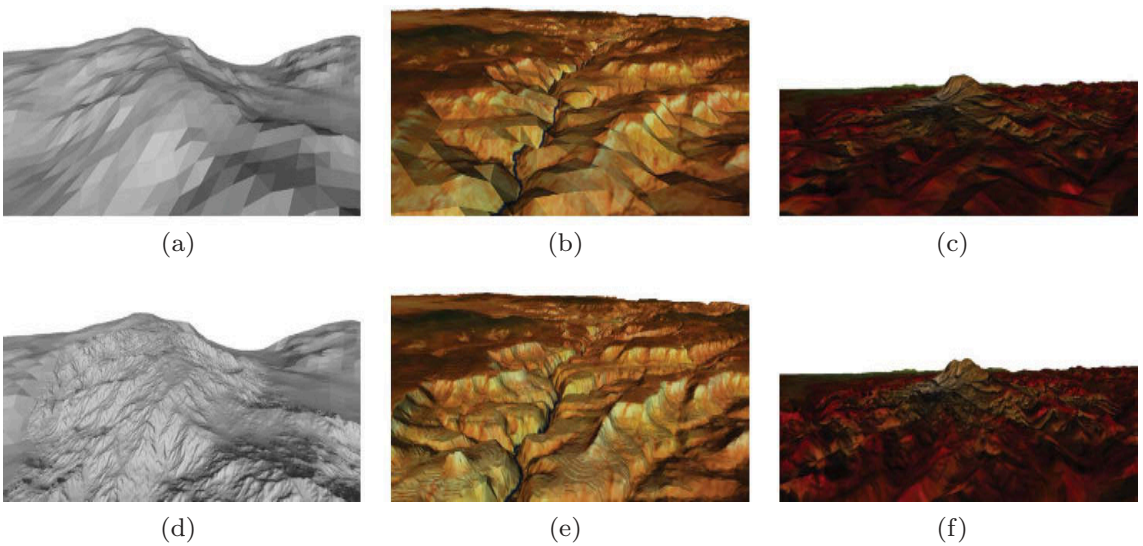


Figure 13. Sample models used in the tests. (a) and (d) Alpine dataset: Grid and hybrid models. (b) and (e) GCanyon dataset: Grid and hybrid models. (c) and (f) PSound dataset: Grid and hybrid models.

rendering management. All the tests were performed on a standard PC system equipped with a Pentium IV 3 GHz processor and a Nvidia GeForce GTX280 graphic card with 512 MB of video memory.

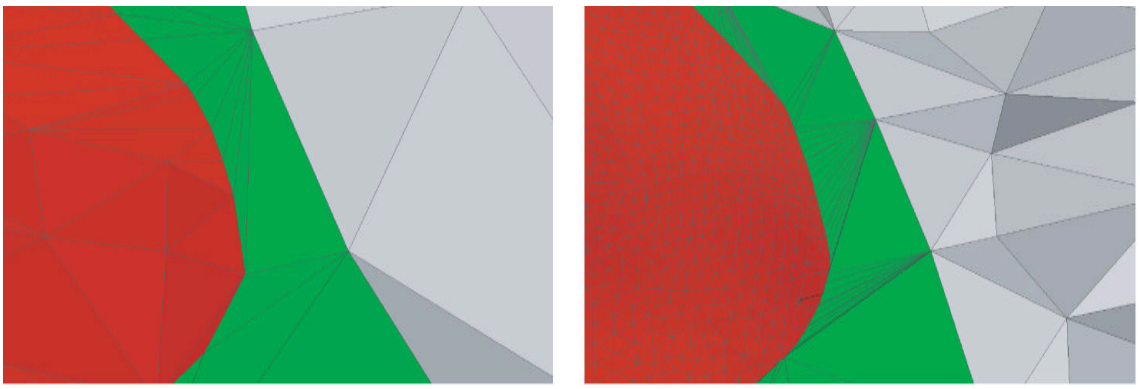
Three sample models, depicted in Figure 13, have been used in the tests. These models, synthetically generated from high-resolution terrain DEMs, present the typical scenario considered in our approach: a hybrid model formed by a base regular grid and a very highly detailed TIN mesh, covering a extension around the 20–30% of the total model area. The original datasets for the GCanyon and PSound models were obtained from the Georgia Tech repository (Turk and Mullins 1998) and the Alpine dataset from Viewfinder Panoramas DEM site (DeFerranti 2005). The number of vertices in the grid and TIN parts of each model can be found in Table 2. Note that all the TIN meshes have a much higher number of vertices than their respective base grids.

The multiresolution method used in the grid mesh is a simple quad-tree based approach derived from Röttger *et al.* (1998). Regarding the TIN mesh, our application employs the View-Dependent Progressive Meshes (VDPM) framework (Hoppe 1998). VDPM is an effective, well-known approach for the LOD rendering of irregular meshes with several different implementations, such as Kim and Lee (2001) or Pajarola and DeCoro (2004). Furthermore, the VDPM method has been adapted to parallel graphic hardware in Hu *et al.* (2009). Since an implementation of the EHM algorithm running in the GPU (Graphics Processing Unit) is planned as future work, this was the main reason for choosing the

Table 2. Sample models information.

Model	Grid #vertices	TIN #vertices	TIN #triangles
Alpine	4225	193586	385339
GCanyon	16641	34331	68032
PSound	16641	54462	108009

Note: TIN, triangulated irregular network.



(a) (b)

Figure 14. Detail of the tessellated area joining TIN and grid meshes. (a) Coarse detailed model. (b) Finer detailed model.

VDPM method. The implementation by Kim and Lee was used since the source code is openly available.

Good quality results were attained in all our tests. The EHM algorithm generated an adaptive tessellation of the area between both meshes boundaries without any visible problems, such as holes or cracks. By way of example, Figure 14 shows a detail of the dynamic tessellation (colored in green) generated during the interactive rendering of a hybrid model at two different resolutions. As can be seen in the figure, the tessellation is well adapted to the meshes and, moreover, there is no evidence of reduced quality in the border of the TIN mesh, caused by the modifications introduced in the multiresolution data generation to comply with the EHM preconditions.

The EHM approach is used to dynamically refine both grid and TIN meshes. The obtained tessellation is generated according to the view conditions of the scene and to the overall LOD of the model. In this way, the LOD of the tessellation is coherent with the resolution of both meshes, presenting a tessellation with an adequate density and proportionally sized triangles. By way of example, Figure 15 shows the variation in the size of the tessellation triangles depending on the LOD selected on the grid and TIN meshes. Much larger and thinner triangles are generated when using a high resolution version of a TIN with a coarse detailed grid in Figure 15b (HM approach), while more reasonable sized triangles are obtained with the view-dependent rendering of the TIN in Figure 15c (EHM approach).

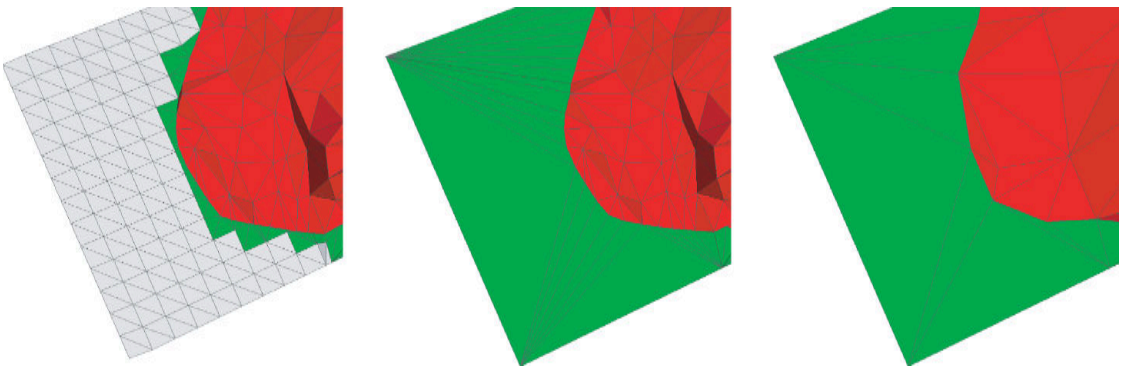
The results obtained during our tests are shown in Tables 3 and 4. Table 3 summarizes the number of primitives generated during the rendering. Each sample model has been tested in three different view configurations, labeled as High, Medium, and Low, using around 70%, 25%, and 5% of the original model triangles, respectively. The three columns in the table contain the average number of triangles used in the grid, TIN and EHM component meshes. Here, we should stress that, using the EHM method, it is possible to render the whole hybrid model using the 5% of the original grid and TIN triangles. The small overhead introduced by the EHM algorithm, in terms of additional primitives included during the hybrid model rendering, is also verified in this table. Note that only around 2100 primitives were needed by the EHM algorithm to join the grid and TIN meshes in the worst case, which corresponded to the fully detailed configuration of the Alpine model. Compared to the complete size of the model, this number is almost negligible and is a reasonable trade off for obtaining well-formed hole-free hybrid meshes.

Table 3. Render primitives used during the visualization of sample models using the EHM algorithm.

Model		grid #tri	TIN #tri	EHM #tri
Alpine	Full	8192	385339	2114
	High	5803	271994	1705
	Med	2016	96609	1223
	Low	396	20740	1109
GCanyon	Full	32768	68032	980
	High	22588	48100	791
	Med	7909	17067	628
	Low	1495	3473	569
PSound	Full	32768	108009	1200
	High	23646	74412	1021
	Med	8023	28355	786
	Low	1913	5248	682

Note: TIN, triangulated irregular network; EHM, extended hybrid meshing.

Table 4 shows the frames-per-second (FPS) obtained during the visualization of the test models using the EHM algorithm. The performance of the EHM algorithm in the High, Medium, and Low view configurations are compared with the performance obtained using the HM algorithm with the full-resolution TIN. The first column shows the average FPS obtained during visualization. The second and third columns show the fraction of time spent in the generation of the view-dependent TIN mesh, and in the tessellation of the PC cells. It is patent visible that the performance overhead introduced by the EHM tessellation algorithm in our experiments is small, representing only a 2.62% of the frame rendering time, in the worst case. The overall gain of the EHM algorithm is highly significant when a low-resolution TIN is extracted, for example, a 4x speed up is obtained for the PSound model in the best case. In fact, since the TIN refinement operation (Kim and Lee 2001) is the slowest stage in the rendering pipeline and represents the performance bottleneck of the system, even better overall results could be achieved using a different implementation. Pajarola and DeCoro's (2004) implementation, for example, is reported to be five times faster than Kim and Lee's one on a slower machine.



(a) (b) (c)

Figure 15. Reduction of tessellation triangles using multiresolution TINs. (a) Detailed grid and TIN meshes. (b) Coarse grid and fine TIN meshes. (c) Coarse grid and TIN meshes.

Table 4. Performance results obtained during the visualization of sample models using the original HM and the EHM algorithm.

Model		FPS	TIN gen time %	EHM gen time %
Alpine	Full TIN	25.54	–	–
	High	4.0	88.87	0.14
	Med	13.5	89.65	0.44
	Low	87.5	83.81	2.07
GCanyon	Full TIN	131.8	–	–
	High	42.8	89.65	0.72
	Med	115.8	83.30	1.28
	Low	469.4	71.62	2.62
PSound	Full TIN	89.0	–	–
	High	23.6	93.17	0.48
	Med	68.8	86.99	1.20
	Low	373.8	75.32	2.48

Note: FPS, frames-per-second; TIN, triangulated irregular network; EHM, extended hybrid meshing.

In summary, the EHM algorithm presented in this article is a flexible and efficient framework for the real-time visualization of hybrid terrain models. The test results obtained show that the EHM algorithm performs high-quality rendering of complex multiresolution hybrid models involving regular and irregular meshes, in an efficient way and without a heavy performance penalty.

6. Conclusions

In this article we have presented a new method for rendering multiresolution hybrid terrain models. Our work is based on the HM algorithm, a previously developed approach for hybrid representation of terrains. The original HM algorithm supports a multiresolution method in the regular base mesh of the model, while the finely detailed TIN parts are statically rendered. The new EHM algorithm succeeds in enhancing the original algorithm by adding view-dependent rendering of the TIN meshes, in a simple and lightweight manner.

The method we present works by constraining the simplification order of the TB vertices, so that tessellation of the region between the meshes boundaries can be performed, regardless of the simplification step of the TIN. These constraints are fairly straightforward to meet, and are enforced during the construction of the view-dependent refinement data structures. Since all these computations take place in the preprocessing phase, no additional time-demanding tasks are performed during real-time visualization. Thanks to this feature, the benefits of our approach are clear, as the whole multiresolution hybrid model can be rendered without incurring in supplementary performance penalties to deal with the varying geometry of multiresolution models. Thus, we have developed and tested a complete framework for the full rendering of multiresolution hybrid terrain models. Using this approach high-quality meshes are obtained, while performance allows for practical applications, as has been shown by the results.

Future work will include the integration of more GPU-friendly multiresolution algorithms into the grid and TIN models, together with an implementation of the algorithm

for leveraging the GPU-processing capabilities. Other improvements to be considered are the minimization or removal of the preconditions during the construction of the TIN multiresolution hierarchy, and the extension of the algorithm to dynamically change the position of the TIN mesh within the base model surface.

Acknowledgments

This work was supported in part by the Xunta de Galicia under projects 08TIC001206PR and 08SIN011291PR and the Program for Consolidation of Competitive Research Groups ref. 2010/06 and 2010/28, and by the Ministry of Science and Innovation of Spain, cofunded by the FEDER funds of the European Union, under contracts TIN2010-16735 and TIN2010-17541. M. Bóo, M. Amor, and J.D. Bruguera are members of the European Network of Excellence on High Performance and Embedded Architecture and Compilation (HiPEAC).

References

Baumann, K., *et al*. 1999. A hybrid, hierarchical data structure for real-time terrain visualization. *In*: B. Werner, ed. *Proceedings of the international conference on computer graphics (CGI '99)*, 7–11 June 1999 Canmore, AB, Canada. Washington: IEEE Computer Society, 85–92.

Bóo, M. and Amor, M., 2009. Dynamic hybrid terrain representation based on convexity limits identification. *International Journal of Geographical Information Science*, 23 (4), 417–439.

Bóo, M., Amor, M., and Döllner, J., 2007. Unified hybrid terrain representation based on local convexifications. *GeoInformatica*, 11 (3), 331–357.

Chai, J., Miyoshi, T., and Nakamae, E., 1998. Contour interpolation and surface reconstruction of smooth terrain models. *In*: D.S. Ebert, H. Rushmeier and H. Hagen, eds. *Proceedings of the conference on visualization (VIS '98)*, 18–23 October 1998 Research Triangle Park, NC, USA. Los Alamitos, CA, USA: IEEE Computer Society, 27–33.

DeFerranti, J., 2005. *Digital elevation models in viewfinder panoramas* [online]. Available from: http://www.viewfinderpanoramas.org [Accessed September 2010].

DeFloriani, L., Marzano, P., and Puppo, E., 1996. Multiresolution models for topographic surface description. *The Visual Computer*, 12 (7), 317–345.

Duchaineau, M., *et al*., 1997. Roaming terrain: real-time optimally adapting meshes. *In*: *Proceedings of the conference on visualization (VIS '97)*, 19–24 October 1997 Phoenix, AZ, USA. Los Alamitos, CA, USA: IEEE Computer Society, 81–88.

El-Sana, J. and Varshney, A., 1999. Generalized view-dependent simplification. *Computer Graphics Forum*, 18 (3), 83–94.

Fuchs, H. and Kedem, Z., 1977. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20 (10), 693–702.

Gröger, G., *et al*., 2008. *Opengis city geography markup language (CityGML) encoding standard* [online]. Available from: http://www.citygml.org/ [Accessed October 2010].

Hoppe, H., 1997. View-dependent refinement of progressive meshes. *In*: L. Pocock, R. Hopkins, D. Ebert and J. Crow, eds. *Proceedings of the 24th annual conference on computer graphics and interactive techniques (SIGGRAPH '97)*, 3–8 August 1997 Los Angeles, CA, USA. New York: ACM Press/Addison-Wesley Publishing Co., 189–198.

Hoppe, H., 1998. Smooth view-dependent level-of-detail control and its application to terrain rendering. *In*: D.S. Ebert, H. Rushmeier and H. Hagen, eds. *Proceedings of the conference on visualization (VIS '98)*, 18–23 October 1998 Research Triangle Park, NC, USA. Los Alamitos, CA, USA: IEEE Computer Society, 35–42.

Hormann, K., Spinello, S., and Schröder, P., 2003. C1-Continuous terrain reconstruction from sparse contours. *In*: T. Ertl, ed. *Proceedings of the vision, modeling, and visualization conference 2003 (VMV '03)*, 19–21 November 2003 München, Germany. Heidelberg, Germany: Akademische Verlagsgesellschaft Aka, 289–297.

Hu, L., Sander, P.V., and Hoppe, H., 2009. Parallel view-dependent refinement of progressive meshes. *In*: E. Haines, M. McGuire, D.G. Aliaga, M.M. Oliveira and S.N. Spencer, eds. *Proceedings of the 2009 symposium on interactive 3D graphics and games (SI3D '09)*, 27 February–1 March 2009 Boston, MA, USA. New York: ACM Press, 169–176.

Kim, J. and Lee, S., 2001. Truly selective refinement of progressive meshes. *In*: *Proceedings of the conference on graphics interface (GRIN '01)*, 7–9 June 2001 Ottawa, ON, Canada. Toronto: Canadian Information Processing Society, 101–110.

Levenberg, J., 2002. Fast view-dependent level-of-detail rendering using cached geometry. *In: Proceedings of the conference on visualization (VIS '02)*, 27 October–01 November 2002 Boston. Washington: IEEE Computer Society, 259–266.

Luebke, D., *et al*., 2002. *Level of detail for 3D graphics*. Burlington, MA, USA: Morgan Kaufmann.

Pajarola, R., 1998. Large scale terrain visualization using the restricted quadtree triangulation. *In*: D.S. Ebert, H. Rushmeier and H. Hagen, eds. *Proceedings of the conference on visualization (VIS '98)*, 18–23 October 1998 Research Triangle Park, NC, USA. Los Alamitos, CA, USA: IEEE Computer Society, 19–26.

Pajarola, R. and DeCoro, C., 2004. Efficient implementation of real-time view-dependent multiresolution meshing. *IEEE Transactions on Visualization and Computer Graphics*, 10 (3), 353–368.

Pajarola, R. and Gobbetti, E., 2007. Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer*, 23 (8), 583–605.

Paredes, E.G., *et al*., 2009. Hybrid terrain visualization based on local tessellations. *In*: A. Ranchordas, J. Pereira and P. Richard, eds. *Proceedings of the fourth international conference on computer graphics theory and applications (GRAPP '09)*, Lisboa, Portugal, 5–8 February 2009 Lisboa, Portugal. Setubal, Portugal: INSTICC Press, 64–69.

Röttger, S., *et al*., 1998. Real-time generation of continuous levels of detail for height fields. *In*: V. Skala, ed. *Proceedings of the international conference in central Europe on computer graphics and visualization (WSCG '98)*, Pilsen, Czech Republic, 9–13 February 1998. Pilsen, Czech Republic: University of West Bohemia, 315–322.

Schneider, J. and Westermann, R., 2006. GPU-friendly high-quality terrain rendering. *In*: V. Skala, ed. *Proceedings of the 14th international conference in central Europe on computer graphics, visualization and computer vision (WSCG '06)*, Pilsen, Czech Republic, 30 January–3 February 2006 Pilsen, Czech Republic. Pilsen, Czech Republic: University of West Bohemia, 49–56.

Turk, G. and Mullins, B., 1998. *Large geometric models archive at Georgia Institute of Technology* [online]. Available from: http://www.cc.gatech.edu/projects/large_models/ [Accessed September 2010].

Yang, B., Shi, W., and Li, Q., 2005. An integrated TIN and grid method for constructing multi-resolution digital terrain models. *International Journal of Geographical Information Science*, 19 (10), 1019–1038.

Ylmaz, T., Güdükbay, U., and Akman, V., 2004. Modeling and visualization of complex geometric environments. *In*: M. Sarfraz, ed. *Geometric modeling: techniques, applications, systems and tools*. Norwell, MA, USA: Kluwer Academic Publishers, 4–30.